



# UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
United States Patent and Trademark Office  
Address: COMMISSIONER FOR PATENTS  
P.O. Box 1450  
Alexandria, Virginia 22313-1450  
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/066,362	01/31/2002	Julian S. Taylor	SMQ-119/P6157	2482
959	7590	12/27/2005	EXAMINER	
LAHIVE & COCKFIELD, LLP. 28 STATE STREET BOSTON, MA 02109			HICKS, MICHAEL J	
			ART UNIT	PAPER NUMBER
			2165	

DATE MAILED: 12/27/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

**Office Action Summary**

Application No.

10/066,362

Applicant(s)

TAYLOR, JULIAN S.

Examiner

Michael J. Hicks

Art Unit

2165

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

**Period for Reply**

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

**Status**

- 1) ☒ Responsive to communication(s) filed on 1/31/2002.
- 2a) ☐ This action is **FINAL**. 2b) ☒ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

**Disposition of Claims**

- 4) ☒ Claim(s) 1-63 is/are pending in the application.
- 4a) Of the above claim(s) \_\_\_\_\_ is/are withdrawn from consideration.
- 5) ☒ Claim(s) 2-6, 10, 21, 29-30, 33, 41-45, 49, 60 and 62 is/are allowed.
- 6) ☒ Claim(s) 1, 7-9, 11-20, 22-28, 31-32, 34-40, 46-48, 50-59, 61, and 63 is/are rejected.
- 7) ☒ Claim(s) 50 is/are objected to.
- 8) ☐ Claim(s) \_\_\_\_\_ are subject to restriction and/or election requirement.

**Application Papers**

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☒ The drawing(s) filed on 31 January 2002 is/are: a) ☒ accepted or b) ☐ objected to by the Examiner.  
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).  
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

**Priority under 35 U.S.C. § 119**

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
- a) ☐ All b) ☐ Some \* c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
2. ☐ Certified copies of the priority documents have been received in Application No. \_\_\_\_\_.
3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- \* See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

- 1) ☒ Notice of References Cited (PTO-892)
- 2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
- 3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)  
Paper No(s)/Mail Date 10/4/2002.
- 4) ☐ Interview Summary (PTO-413)  
Paper No(s)/Mail Date. \_\_\_\_\_.
- 5) ☐ Notice of Informal Patent Application (PTO-152)
- 6) ☐ Other: \_\_\_\_\_.

### **DETAILED ACTION**

1. Claims 1-63 are pending in the instant application.

#### ***Claim Objections***

2. Claim 50 objected to because of the following informalities:

Claim 50 is dependent upon Claim 56, which is improper as a claim may only depend from a preceding claim. For the purpose of further examination, it will be assumed that Claim 50 is dependent upon Claim 46.

Appropriate correction is required.

#### ***Claim Rejections - 35 USC § 112***

3. The following is a quotation of the second paragraph of 35 U.S.C. 112:

The specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention.

4. Claim 20 rejected under 35 U.S.C. 112, second paragraph, as being indefinite for failing to particularly point out and distinctly claim the subject matter which applicant regards as the invention. Claim 20 recites the limitation "the first computer readable medium" in line 4. There is insufficient antecedent basis for this limitation in the claim.

#### ***Claim Rejections - 35 USC § 103***

5. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

Art Unit: 2165

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

6. Claims 1, 7-9, 14, 16-17, 19-20, 22-23, 26-28, 31-32, 35, 37-40, 46-48, 53, 55-56, 58-59, and 61 rejected under 35 U.S.C. 103(a) as being unpatentable over Vishlitzky et al. (U.S. Patent Number 6,145,006 and referred to hereinafter as Vishlitzky) in view of Schwan et al. ("Improving Performance by Use of Adaptive Objects: Experimentation with a Configurable Multiprocessor Thread Package", Proceedings the 2nd International Symposium on High Performance Distributed Computing, 1993., 20-23 Jul 1993; Pgs. 59-66 and referred to hereinafter as Schwan).

As per Claims 1, 20, 28, 40, and 59 Vishlitzky discloses a method, system and computer readable medium for implementing a locking mechanism to control access to a shared resource (i.e. *"...a locking mechanism is provided within storage system which allows an attached host to lock particular storage system resources or the entire storage system. The lock mechanism is recognized and usable by heterogeneous hosts attached to the storage system...With the lock mechanism of the present invention, cross-platform coordination of storage system operations may be achieved."* The preceding text excerpt clearly indicates that the invention presents a locking mechanism to control access to a shared resource.) (Column 4, Lines 49-59), comprising: a file system (i.e. *"Each of these host computers may at some point in time have access to the same logical drives within the storage system."* The preceding text excerpt clearly indicates that a least one logical drive/file system exists in the system.) (Column 1, Lines 39-41); a processor in communication with the first computer readable medium and file system (i.e. *"The storage system resources are each accessible by a plurality of heterogeneous host computers coupled to the storage system."* The preceding text excerpt clearly indicates that there is at least one host computer/processor which is

Art Unit: 2165

coupled to/in communication with the storage system/logical drive/file system.) (Column 2, Lines 5-7);

code implemented in the computer readable medium executed by the processor to

cause the processor to perform: (i) receiving a request to access the shared resource

(i.e. *"The first of the four lock commands is the LOCK command. This command is issued by a host, via a write buffer transaction described above, in order to lock a resource of storage system."* The preceding

text excerpt clearly indicates that a command/request to access and lock a resource of the storage

system/shared resource is issued and received.) (Column 7, Lines 66-67; Column 8, Lines 1-2); (ii)

determining whether a first file in the file system has a first name (i.e. *"The storage system*

*will either grant the lock if the lock is not already outstanding, or will deny the request...if the lock was set*

*the ASCII value stored...might be "LOCKED" or some other similar ASCII string. Similarly if the lock was*

*not set the ASCII value might be "UNLOCKED".*" The preceding text excerpt clearly indicates that a field

in a data structure/filename is checked for a value indicative of a lock status. Note that while the field

being checked for the ASCII string indicating "LOCKED"/first name is not a filename, it is a field in a data

structure that is meant to be accessed in much the same way as a filename. Because both the field and

the filename may be accessed through platform commands which are common and exist in most

platforms, the field in the data structure and the filename are considered to be functionally equivalent.)

(Column 5, Lines 45-52; Column 8, Lines 4-6); (iii) renaming the first file to a second name if the

first file has the first name (i.e. *The storage system will either grant the lock if the lock is not already*

*outstanding, or will deny the request...if the lock was set the ASCII value stored...might be "LOCKED" or*

*some other similar ASCII string. Similarly if the lock was not set the ASCII value might be "UNLOCKED".*"

The preceding text excerpt clearly indicates if the lock is available for the request the lock will be granted

and the ASCII string/filename will be updated/changed/renamed accordingly.) (Column 5, Lines 45-52;

Column 8, Lines 4-6); and (v) renaming the first file to the first name after the second file is

updated (i.e. *"The second lock command is the UNLOCK Command. As may be expected, this*

*command is used by an application to release a previously set lock of a storage system resource...if the*

Art Unit: 2165

*lock was set the ASCII value stored...might be "LOCKED" or some other similar ASCII string. Similarly if the lock was not set the ASCII value might be "UNLOCKED".* The preceding text excerpt clearly indicates that after the second file is updated, the lock will be released and the ASCII value/filename of the field/first file will be updated accordingly.) (Column 5, Lines 45-52; Column 8, Lines 25-27).

Vishlitzky fails to disclose updating a second file in the file system to indicate the received request in a queue of requests to the shared resource if the first file is renamed to the second name, wherein an ordering of the requests in the queue is used to determine whether access to the shared resource is granted to the request.

Schwan discloses updating a second file in the file system to indicate the received request in a queue of requests to the shared resource if the first file is renamed to the second name, wherein an ordering of the requests in the queue is used to determine whether access to the shared resource is granted to the request (i.e. "...global ordering is strictly maintained in the work sharing queue...If lock status indicates that the thread must wait for the lock, then the waiting method is determined by the acquisition module of the lock object...The last part of the lock object's policy is the release module, which selects the next thread that is granted access to the lock." The preceding text excerpt clearly indicates that a work sharing queue/queue of requests is present which determines the order of processes to be granted locks (e.g. whether or not access to the shared resource is granted to the request). Note that the work sharing queue qualifies as a lockable resource, and can therefore be managed with the renaming method as above.) (Page 63, Column 1, Paragraph 1; Page 65, Column 1, Paragraph 3-4).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of Schwan to include updating a second file in the file system to indicate the received request in a queue of requests to the shared resource if the first file is renamed to the second name, wherein

Art Unit: 2165

an ordering of the requests in the queue is used to determine whether access to the shared resource is granted to the request with the motivation of using kernel abstractions to demonstrate the advantages of adaptive locks and presenting a structure for adaptive objects which can be used to build different operating system abstractions for parallel and distributed systems (Schwan, Page 59, Column 1; Page 60, Column 2).

As per Claims 7, 22, 31, 46, and 61 Vishlitzky fails to disclose a lease data structure indicates at least one request indicated in the queue in the second file granted access to the shared resource and, for each request granted access to the shared resource, a lease time during which the request is granted access to the shared resource, further comprising: determining whether one request in the queue is permitted access to the shared resource based on the ordering of the requests in the queue; updating the lease data structure to indicate the determined request and the lease time if the determined request is permitted access to the shared resource; and returning a message to the determined request indicating that access to the shared resource is granted and the lease time during which access is granted.

Schwan discloses a lease data structure indicates at least one request indicated in the queue in the second file granted access to the shared resource and, for each request granted access to the shared resource, a lease time during which the request is granted access to the shared resource (i.e. *"In the above algorithm, Waiting-Threshold specifies the threshold of the number of waiting threads for a lock which determines when the number of spins should be raised and n is a lock-specific constant. The threshold and n are different for different locks and depend on the locking pattern and the length of the critical section."* The preceding text excerpt

Art Unit: 2165

clearly indicates that a data structure exists to track the number of waiting threads and a granted request. Depending on the number of waiting threads and the length of the critical section of the granted request, a number of spins/lease time is granted to the current request.) (Page 62, Column 2, Paragraph 3), further comprising: determining whether one request in the queue is permitted access to the shared resource based on the ordering of the requests in the queue (i.e. "...*global ordering is strictly maintained in the work sharing queue...If lock status indicates that the thread must wait for the lock, then the waiting method is determined by the acquisition module of the lock object...The last part of the lock object's policy is the release module, which selects the next thread that is granted access to the lock.*" The preceding text excerpt clearly indicates that requests in the queue are permitted access to a shared resource based on the ordering in the queue (e.g. when one request releases a lock, it can discern from the queue, which request to give the lock to).) (Page 63, Column 1, Paragraph 1; Page 65, Column 1, Paragraph 3-4); updating the lease data structure to indicate the determined request and the lease time if the determined request is permitted access to the shared resource (i.e. "*The constants Waiting-Threshold and n need to be varied in order to attain the optimal adaptation policy for each specific lock...At this time, attribute information like thread-id, priorities, ownership, etc. is processed by the lock's policy.*" The preceding text excerpt clearly indicates that when the number of waiting threads changed (e.g. when a request is permitted access to a shared resource) the number of spins/lease time is varied/updated and that attribute information including thread id is present.) (Page 62, Column 2, Paragraph 3; Page 65, Column 1, Paragraph 3); and returning a message to the determined request indicating that access to the shared resource is granted and the lease time during which access is granted (i.e. "*If locks status indicates that the thread must wait for the lock, then the waiting method is determined by the acquisition module of the lock object...The last part of the lock object's policy is the release module, which selects the next thread that is granted access to the lock.*" The preceding text excerpt clearly indicates that the request is



Art Unit: 2165

informed when it is granted access to the shared resource, and is aware of the current number of spins/waiting time/lease time.) (Page 65, Column 1, Paragraphs 3-4).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of Schwan to include a lease data structure indicates at least one request indicated in the queue in the second file granted access to the shared resource and, for each request granted access to the shared resource, a lease time during which the request is granted access to the shared resource, further comprising: determining whether one request in the queue is permitted access to the shared resource based on the ordering of the requests in the queue; updating the lease data structure to indicate the determined request and the lease time if the determined request is permitted access to the shared resource; and returning a message to the determined request indicating that access to the shared resource is granted and the lease time during which access is granted with the motivation of using kernel abstractions to demonstrate the advantages of adaptive locks and presenting a structure for adaptive objects which can be used to build different operating system abstractions for parallel and distributed systems (Schwan, Page 59, Column 1; Page 60, Column 2).

As per Claims 8, 23, 32, and 47, Vishlitzky fails to disclose the request to access is for exclusive access to the shared resource, wherein determining whether one request in the queue is permitted access to the shared resource further comprises: determining the request following the request at a top of the queue after determining that the lease time has expired; removing the request at the top of the queue; updating the queue to indicate the determined request at the top of the queue; and updating the

Art Unit: 2165

lease data structure to identify the determined request and set a new lease time for the determined request in the lease data structure during which the request has exclusive access to the shared resource.

Schwan discloses the request to access is for exclusive access to the shared resource, wherein determining whether one request in the queue is permitted access to the shared resource further comprises (i.e. *"If locks status indicates that the thread must wait for the lock, then the waiting method is determined by the acquisition module of the lock object...The last part of the lock object's policy is the release module, which selects the next thread that is granted access to the lock."*) The preceding text excerpt clearly indicates that the request is for an exclusive lock, due to the fact that the lock is only granted after a thread/request has released it.) (Page 65, Column 1, Paragraphs 3-4); determining the request following the request at a top of the queue after determining that the lease time has expired (i.e. *"...global ordering is strictly maintained in the work sharing queue...If lock status indicates that the thread must wait for the lock, then the waiting method is determined by the acquisition module of the lock object...The last part of the lock object's policy is the release module, which selects the next thread that is granted access to the lock."*) The preceding text excerpt clearly indicates that the request which is next in line in the scheduling queue is selected/determined after the number of spins/lease time has expired.) (Page 63, Column 1, Paragraph 1; Page 65, Column 1, Paragraph 3-4); removing the request at the top of the queue (i.e. *"...global ordering is strictly maintained in the work sharing queue..."*) The preceding text excerpt clearly indicates that once a thread/request has released a lock, it is removed from the top of the queue.) (Page 63, Column 1, Paragraph 1); updating the queue to indicate the determined request at the top of the queue (i.e. *"...global ordering is strictly maintained in the work sharing queue... The last part of the lock object's policy is the release module, which selects the next thread that is granted access to the lock."*) The preceding text excerpt clearly indicates that the queue is updated to indicate that the thread/request which is granted the lock is placed at the head/top of the queue.) (Page 63, Column 1,

Paragraph 1; Page 65, Column 1, Paragraph 4); and updating the lease data structure to identify the determined request and set a new lease time for the determined request in the lease data structure during which the request has exclusive access to the shared resource (i.e. *"The constants Waiting-Threshold and n need to be varied in order to attain the optimal adaptation policy for each specific lock...At this time, attribute information like thread-id, priorities, ownership, etc. is processed by the lock's policy."* The preceding text excerpt clearly indicates that when the number of waiting threads changed (e.g. when a request is permitted access to a shared resource) the number of spins/lease time is varied/updated to indicate the amount of time the thread/request which was granted the lock will hold the lock, and that attribute information including thread id is also indicated.) (Page 62, Column 2, Paragraph 3; Page 65, Column 1, Paragraph 3).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of Schwan to include the request to access is for exclusive access to the shared resource, wherein determining whether one request in the queue is permitted access to the shared resource further comprises: determining the request following the request at a top of the queue after determining that the lease time has expired; removing the request at the top of the queue; updating the queue to indicate the determined request at the top of the queue; and updating the lease data structure to identify the determined request and set a new lease time for the determined request in the lease data structure during which the request has exclusive access to the shared resource with the motivation of using kernel abstractions to demonstrate the advantages of adaptive locks and presenting a structure for adaptive objects which can be used to build different operating system

Art Unit: 2165

abstractions for parallel and distributed systems (Schwan, Page 59, Column 1; Page 60, Column 2).

As per Claims 9 and 48, Vishlitzky fails to disclose the queue and lease structure are updated after renaming the first file from the first name to the second name.

Schwan discloses the queue and lease structure are updated after renaming the first file from the first name to the second name (i.e. "...global ordering is strictly maintained in the work sharing queue...If lock status indicates that the thread must wait for the lock, then the waiting method is determined by the acquisition module of the lock object...The last part of the lock object's policy is the release module, which selects the next thread that is granted access to the lock." The preceding text excerpt clearly indicates that the work sharing queue is updated. Note that the lease data structure is only updated when the queue is updated. Also note that the work sharing queue qualifies as a lockable resource, and can therefore be managed with the renaming method as above.) (Page 63, Column 1, Paragraph 1; Page 65, Column 1, Paragraph 3-4).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of Schwan to include the queue and lease structure are updated after renaming the first file from the first name to the second name with the motivation of using kernel abstractions to demonstrate the advantages of adaptive locks and presenting a structure for adaptive objects which can be used to build different operating system abstractions for parallel and distributed systems (Schwan, Page 59, Column 1; Page 60, Column 2).

As per Claims 14, 35, and 53, Vishlitzky fails to disclose returning a retry message to the request indicating to retry the request to access the shared resource

and the lease time if the request is determined not to be permitted access to the shared resource, wherein the request is retried after the lease time has expired.

Schwan discloses returning a retry message to the request indicating to retry the request to access the shared resource and the lease time if the request is determined not to be permitted access to the shared resource, wherein the request is retried after the lease time has expired (i.e. *"If locks status indicates that the thread must wait for the lock, then the waiting method is determined by the acquisition module of the lock object...The last part of the lock object's policy is the release module, which selects the next thread that is granted access to the lock."*)

The preceding text excerpt clearly indicates that if the thread/is request is not granted access to the shared resource, it must wait until the number of spins/lease time has expired, which is indicated by the acquisition module of the lock object. After the number of spins/lease time has expired it will be granted access if it is at the top of the queue.) (Page 65, Column 1, Paragraphs 3-4)

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of Schwan to include returning a retry message to the request indicating to retry the request to access the shared resource and the lease time if the request is determined not to be permitted access to the shared resource, wherein the request is retried after the lease time has expired with the motivation of using kernel abstractions to demonstrate the advantages of adaptive locks and presenting a structure for adaptive objects which can be used to build different operating system abstractions for parallel and distributed systems (Schwan, Page 59, Column 1; Page 60, Column 2).

As per Claims 16, 26, 37, and 55 Vishlitzky discloses the locking mechanism is capable of being executed on multiple operating system platforms (i.e. *"The lock mechanism is recognized and usable by heterogeneous hosts attached to the storage system...With the lock*

Art Unit: 2165

*mechanism of the present invention, cross-platform coordination of storage system operations may be achieved.*" The preceding text excerpt clearly indicates that the locking mechanism is designed for cross platform/operating system platform coordination/execution.) (Column 4, Lines 49-59), and wherein the steps of renaming and updating correspond to native operating system commands implemented across operating system platforms (i.e. "According to the present invention, the storage system locks described above are manipulated through the use of specialized storage system commands. Generally, the specialized commands are achieved by using standard storage system commands in unique ways to perform functions not necessarily defined by the communications protocols associated with the host computers attached to storage system. For example, for open systems type host computers, the SCSI read buffer and write buffer commands are used in a unique ways to set, release and examine locks." The preceding text excerpt clearly indicates that standard/native storage system commands are used to lock and unlock resources, which includes renaming and updating as described above.) (Column 6, Lines 11-21).

As per Claims 17, 38, and 56, Vishlitzky discloses the locking mechanism is implemented in a cross-platform computer programming language that is called by applications seeking to access the shared resource (i.e. "Here, a set of commands are provided for manipulating the locks and for getting status reports associated with the locks. With such an arrangement, storage system resources may be shared amongst a plurality of heterogeneous host computers even though the different host computers may not communicate with each other." The preceding text excerpt clearly indicates that because the commands are usable by a plurality heterogeneous host computers, the language of implementation must be a cross-platform computer language. Also, because these commands, which constitute the locking mechanism, are used to manipulate the locks, they must, at some point, be called by applications which wish to access the shared resource (e.g. manipulate the locks to acquire access).) (Column 2, Lines 30-35).

As per Claims 19, 27, 39 and 58 Vishlitzky discloses the shared resource comprises a data structure and wherein the access comprises one of read or write accesses to the shared data structure (i.e. *"Each of the disks within a disk array may be considered a storage system resource...the SCSI commands utilized to effect lock operations are the 'write buffer' and 'read buffer' commands."* The preceding text excerpt clearly indicates that a disk (which may be logically considered a data structure) comprises a shared resource, and that the accesses are made using read buffer/read and write buffer/write commands.) (Column 3, Lines 35-37; Column 6, Lines 35-37).

7. Claims 11-13, 24, 34, 50-52, and 63 rejected under 35 U.S.C. 103(a) as being unpatentable over Vishlitzky in view of Schwan in further view of Taterinov et al., ("A semi-Optimistic Database Scheduler Based on Commit Ordering", NDSU Tech Report, 1997 and referred to hereinafter as Tatarinov) and DS ("class readwritelock : public abstractsemaphore", Authored: November 1998; Accessed at: <http://www.ligo-wa.caltech.edu/gds/dtt/dtt/readwritelock.html>; Nov. 28, 2005.).

As per Claims 11, 24, 34, 50, and 63, Vishlitzky fails to disclose the request to access is for non-exclusive access to the shared resource and wherein the lease data structure indicates a number of requests allowed simultaneous access to the shared resource, wherein determining whether one considered request in the queue is permitted access to the shared resource further comprises: determining whether a number of current readers is less than the allowed readers; determining whether less than the number of current readers precedes the considered request in the queue; updating the lease data structure to identify the considered request and set a new lease time for the considered request to have non-exclusive access to the shared resource if the number of current readers is less than the allowed readers and less than the

number of current readers precedes the considered request in the queue; and incrementing the number of current readers after updating the lease data structure to indicate the considered request.

Schwan discloses updating the lease data structure to identify the considered request and set a new lease time for the considered request (i.e. "*In the above algorithm, Waiting-Threshold specifies the threshold of the number of waiting threads for a lock which determines when the number of spins should be raised and n is a lock-specific constant. The threshold and n are different for different locks and depend on the locking pattern and the length of the critical section...global ordering is strictly maintained in the work sharing queue...If lock status indicates that the thread must wait for the lock, then the waiting method is determined by the acquisition module of the lock object...The last part of the lock object's policy is the release module, which selects the next thread that is granted access to the lock.*") The preceding text excerpt clearly indicates that each request, including the considered request, is ordered in the work-sharing queue, and a number of spins/lease time is assigned to it.) (Page 62, Column 2, Paragraph 3; Page 63, Column 1, Paragraph 1; Page 65, Column 1, Paragraph 3-4).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of Schwan to include discloses updating the lease data structure to identify the considered request and set a new lease time for the considered request with the motivation of using kernel abstractions to demonstrate the advantages of adaptive locks and presenting a structure for adaptive objects which can be used to build different operating system abstractions for parallel and distributed systems (Schwan, Page 59, Column 1; Page 60, Column 2).

DS discloses the lease data structure indicates a number of requests allowed simultaneous access to the shared resource (i.e. "*Constructs a read/write lock. Takes the*



Art Unit: 2165

*maximum number of concurrent read locks as argument...*" The preceding text excerpt clearly indicates that the maximum number of readers is specified and could be stored in the lease data structure.) (Page 2, Paragraph 2); determining whether a number of current readers is less than the allowed readers (i.e. *"Multiple read locks (up to maxuse) can be granted)..."* The preceding text excerpt clearly indicates that a check is made before granting the lock to see if the number of readers has reached it's maximum.) (Page 2, Paragraph 6); determining whether less than the number of current readers precedes the considered request in the queue (i.e. *"Multiple read locks (up to maxuse) can be granted)..."* The preceding text excerpt clearly indicates that if a number of readers equaling the maximum number of allowed readers which had priority over the current request existed in the queue, the read lock would not be granted to the current request.) (Page 2, Paragraph 6); the considered request having non-exclusive access to the shared resource if the number of current readers is less than the allowed readers and less than the number of current readers precedes the considered request in the queue (i.e. *"Multiple read locks (up to maxuse) can be granted)..."* The preceding text excerpt clearly indicates if the above two conditions were met, a non-exclusive read lock may be granted.) (Page 2, Paragraph 6).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of DS to include the lease data structure indicates a number of requests allowed simultaneous access to the shared resource and determining whether a number of current readers is less than the allowed readers; determining whether less than the number of current readers precedes the considered request in the queue; and the considered request having non-exclusive access to the shared resource if the number of current readers is less than the allowed readers and less than the number of current readers precedes the considered request

in the queue with the motivation of enable a read lock to be used by multiple readers simultaneously (DS, Page 2, Paragraph 1).

Tatarinov discloses the request to access is for non-exclusive access to the shared resource (i.e. *"When placed in the queue, each read request increments the latch counter."* The preceding text excerpt clearly indicates that the request to access is a read/non-exclusive access request.) (Page 8, Paragraph 2) and wherein determining whether one considered request in the queue is permitted access to the shared resource further comprises: incrementing the number of current readers after updating the lease data structure to indicate the considered request (i.e. *"When placed in the queue, each read request increments the latch counter."* The preceding text excerpt clearly indicates a latch counter indicating the number of current readers in incremented after the queue and lease data structure are updated.) (Page 8, Paragraph 2).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of Tatarinov to include the request to access is for non-exclusive access to the shared resource and wherein determining whether one considered request in the queue is permitted access to the shared resource further comprises: incrementing the number of current readers after updating the lease data structure to indicate the considered request with the motivation of ensuring concurrency control in the database system (Tatarinov, Page 1, Paragraph 2).

As per Claims 12 and 51, Vishlitzky fails to disclose removing one request for non-exclusive access from the queue whose lease time has expired; removing indication from the lease data structure of the request removed from the queue; and decrementing a field in the lease data structure indicating the number of current readers.

Schwan discloses removing one request for non-exclusive access from the queue whose lease time has expired (i.e. "...global ordering is strictly maintained in the work sharing queue...If lock status indicates that the thread must wait for the lock, then the waiting method is determined by the acquisition module of the lock object...The last part of the lock object's policy is the release module, which selects the next thread that is granted access to the lock." The preceding text excerpt clearly indicates that once the request lease time expires, it releases the it's lock and is removed from the work sharing queue.) (Page 63, Column 1, Paragraph 1; Page 65, Column 1, Paragraph 3-4); and removing indication from the lease data structure of the request removed from the queue (i.e. "The constants *Waiting-Threshold* and *n* need to be varied in order to attain the optimal adaptation policy for each specific lock...At this time, attribute information like *thread-id*, *priorities*, *ownership*, *etc.* is processed by the lock's policy." The preceding text excerpt clearly indicates that once the lock has been release, it's attribute information is removed from the lease data structure as well as the queue.) (Page 62, Column 2, Paragraph 3; Page 65, Column 1, Paragraph 3).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of Schwan to include removing one request for non-exclusive access from the queue whose lease time has expired; removing indication from the lease data structure of the request removed from the queue with the motivation of using kernel abstractions to demonstrate the advantages of adaptive locks and presenting a structure for adaptive objects which can be used to build different operating system abstractions for parallel and distributed systems (Schwan, Page 59, Column 1; Page 60, Column 2).

Tatarinov discloses decrementing a field in the lease data structure indicating the number of current readers (i.e. "After a read is performed the counter is decremented." The preceding text excerpt clearly indicates that a latch counter indicating the number of read locks available

Art Unit: 2165

is decremented after removing the indication of the request from the queue and lease data structure.)

(Page 8, Paragraph 2).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of Tatarinov to include decrementing a field in the lease data structure indicating the number of current readers with the motivation of ensuring concurrency control in the database system (Tatarinov, Page 1, Paragraph 2).

As per Claims 13 and 52, Vishlitzky fails to disclose the queue is capable of including entries for exclusive and non-exclusive access to the shared resource, and wherein the lease data structure is updated to indicate the considered request and set a new least time for the considered request to have non-exclusive access to the shared resource if there is no exclusive access request between a top of the queue and the considered request.

Schwan discloses the lease data structure is updated to indicate the considered request and set a new least time (i.e. *"In the above algorithm, Waiting-Threshold specifies the threshold of the number of waiting threads for a lock which determines when the number of spins should be raised and n is a lock-specific constant. The threshold and n are different for different locks and depend on the locking pattern and the length of the critical section. The constants Waiting-Threshold and n need to be varied in order to attain the optimal adaptation policy for each specific lock...At this time, attribute information like thread-id, priorities, ownership, etc. is processed by the lock's policy."* The preceding text excerpt clearly indicates that the lease data structure is updated to indicate the considered request and to set a new number of spins/lease time.) (Page 62, Column 2, Paragraph 3-4; Page 65, Column 1, Paragraph 3).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of Schwan to include the lease data structure is updated to indicate the considered request and set a new least time with the motivation of using kernel abstractions to demonstrate the advantages of adaptive locks and presenting a structure for adaptive objects which can be used to build different operating system abstractions for parallel and distributed systems (Schwan, Page 59, Column 1; Page 60, Column 2).

Tatarinov discloses the queue is capable of including entries for exclusive and non-exclusive access to the shared resource (i.e. *See figure on page 8*. The figure on page 8 clearly indicates that both write/exclusive and read/non-exclusive locks are present in the queue.) (Page 8), and the considered request may have non-exclusive access to the shared resource if there is no exclusive access request between a top of the queue and the considered request (i.e. *"It should set the counter to the number of read requests from the head of the queue to the first write request."* The preceding text excerpt and the figure on page 8 clearly indicates that a read/non-exclusive access request may only be granted if there are no write/exclusive access requests preceding it in the queue.) (Page 8; Page 9, Paragraph 3).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of Tatarinov to include the queue is capable of including entries for exclusive and non-exclusive access to the shared resource and the considered request may have non-exclusive access to the shared resource if there is no exclusive access request between a top of the queue and the considered request with the motivation of ensuring concurrency control in the database system (Tatarinov, Page 1, Paragraph 2).

8. Claims 15, 25, 36, and 54 rejected under 35 U.S.C. 103(a) as being unpatentable over Vishlitzky in view of Krieger et al. ("A Fair, Fast Scalable Reader-Writer Lock", Proceedings of the International Conference on Parallel Processing, 1993 and referred to hereinafter as Krieger).

As per Claims 15, 25, 36, and 54, Vishlitzky discloses determining whether the first file has the first name (i.e. *"The storage system will either grant the lock if the lock is not already outstanding, or will deny the request...if the lock was set the ASCII value stored...might be "LOCKED" or some other similar ASCII string. Similarly if the lock was not set the ASCII value might be "UNLOCKED"."*) The preceding text excerpt clearly indicates that a field in a data structure/filename is checked for a value indicative of a lock status. Note that while the field being checked for the ASCII string indicating "LOCKED"/first name is not a filename, it is a field in a data structure that is meant to be accessed in much the same way as a filename. Because both the field and the filename may be accessed through platform commands which are common and exist in most platforms, the field in the data structure and the filename are considered to be functionally equivalent.); renaming the first file to the second name if the first file has the first name (i.e. *The storage system will either grant the lock if the lock is not already outstanding, or will deny the request...if the lock was set the ASCII value stored...might be "LOCKED" or some other similar ASCII string. Similarly if the lock was not set the ASCII value might be "UNLOCKED".*) The preceding text excerpt clearly indicates if the lock is available for the request the lock will be granted and the ASCII string/filename will be updated/changed/renamed accordingly.) (Column 5, Lines 45-52; Column 8, Lines 4-6); and renaming the first file to the first name after the second file is updated (i.e. *"The second lock command is the UNLOCK Command. As may be expected, this command is used by an application to release a previously set lock of a storage system resource...if the lock was set the ASCII value stored...might be "LOCKED" or some other similar ASCII string. Similarly if the lock was not set the ASCII value might be "UNLOCKED"."*) The preceding text excerpt clearly indicates that after the second file is updated, the lock will be released and the ASCII

Art Unit: 2165

value/filename of the field/first file will be updated accordingly.) (Column 5, Lines 45-52; Column 8, Lines 25-27).

Vishlitzky fails to disclose receiving a request to remove an entry from the queue; and updating the second file to indicate that the request is removed from the queue.

Krieger discloses receiving a request to remove an entry from the queue (i.e. "...readerUnlock releases the lock by removing its local structure from the queue." The preceding text excerpt clearly indicates that the readerUnlock sends a request to remove a lock request from the queue.) (Page 3, Column 1, Paragraph 2); and updating the second file to indicate that the request is removed from the queue (i.e. "...the releasing processor can simply dequeue itself by modifying the previous and next fields of it's neighbors in the linked list." The preceding text excerpt clearly indicates that a linked list in the second file is updated to dequeue the request (e.g. indicate that the request is removed from the queue.) (Page 3, Column 1, Paragraph 2).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Vishlitzky with the teachings of Krieger to include receiving a request to remove an entry from the queue and updating the second file to indicate that the request is removed from the queue with the motivation of creating a lock which will scale to large numbers of processors by using only fetch and store operations (Krieger, Page 1, Column 2, Paragraph 1).

9. Claims 18 and 57 rejected under 35 U.S.C. 103(a) as being unpatentable over Vishlitzky in view of Schwan as applied to claims 1, 7-9, 14, 16-17, 19-20, 22-23, 26-28, 31-32, 35, 37-40, 46-48, 53, 55-56, 58-59, and 61 above, and further in view of Judd et al. ("Design issues for efficient implementation of MPI in Java", Proceedings of the ACM 1999

conference on Java Grande, San Francisco, California, United States; Pages: 58 - 65; Year of Publication: 1999 and referred to hereinafter as Judd).

As per Claims 18 and 57, Vishlitzky fails to disclose the cross-platform computer programming language comprises Java.

Judd discloses the cross-platform computer programming language comprises Java (i.e. *"A pure Java implementation is very desirable as it inherits all of Java's cross platform, security, and language safety features."* The preceding text excerpt clearly indicates that Java would be a desirable choice for cross platform computer applications.) (Page 61, Column 1, Paragraph 1).

It would have been obvious to one skilled in the art at the time of Applicants invention to modify the teachings of Judd with the teachings of Schwan to include the cross-platform computer programming language comprises Java with the motivation of Using Java's cross platform features (Judd, Page 61, Column 1, Paragraph 1).

### ***Allowable Subject Matter***

10. Claims 2-6, 10, 21, 29-30, 33, 41-45, 49, 60, and 62 would be allowable if rewritten to overcome the rejection(s) under 35 U.S.C. 112, 2nd paragraph, set forth in this Office action and to include all of the limitations of the base claim and any intervening claims.

11. The following is a statement of reasons for the indication of allowable subject matter: because the prior art matched the claimed inventions functionality in renaming a file to indicate lock status, but did not match doing so in the form of having multiple files, the process of checksumming becomes moot in respect tot the prior art. Due to



Art Unit: 2165

this the indicated claims, which all include the use of checksums, are considered allowable subject matter.

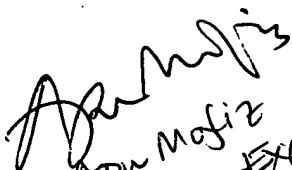
***Points of Contact***

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Michael J. Hicks whose telephone number is (571) 272-2670. The examiner can normally be reached on Monday - Friday 8:30a - 5:00p.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jeffrey Gaffin can be reached on (571) 272-4146. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Michael J Hicks  
Art Unit 2165  
(571) 272-2670

  
Apu Mofiz  
Primary Examiner  
Technology Center 2100